Testing

Contents

[Test Plan 1](#_Toc383184865)

[Test data 2](#_Toc383184866)

[Utilities.js - unit tests 3](#_Toc383184867)

[Ant.js – unit tests 27](#_Toc383184868)

[Nest.js - unit tests 32](#_Toc383184869)

[Worker.js - Integration tests 33](#_Toc383184870)

[Soldier.js – integration tests 36](#_Toc383184871)

[Food.js - integration tests 38](#_Toc383184872)

[FoodSystem.js – unit tests 40](#_Toc383184873)

[NestPiece.js – integration tests 46](#_Toc383184874)

[Pheromone.js – integration tests 47](#_Toc383184875)

[Queen.js – integratoin tests 49](#_Toc383184876)

[Compatibility tests 50](#_Toc383184877)

[Stress tests 55](#_Toc383184878)

[User interface functional testing 59](#_Toc383184879)

[Biological tests 65](#_Toc383184880)

# Test Plan

The strategy chosen to test the solution is to test the basic functionality of functions which are used throughout the application i.e. functions in utilities.js. These will be tested with automatic unit tests which use black box testing to check the output is correct for different inputs. Drawing aspects of the application will be tested by checking that the output is as expected. This will be done for all classes with a \*.draw function. The methods in the ant, worker, soldier and queen classes will be tested with black box unit tests, white box unit tests and integration testing. As the majority of these methods are logic based i.e. they have lots of different conditions rather than calculating a specific thing, it is easiest to test each non obvious path through the method.

Certain modules will not need testing due to their simplicity. The canvas module does not require testing as it is basically a reimplementation of calls to the HTML canvas API so that they can be used in a shorthand version. What’s more the map module only contains very basic functions and so does not need to be tested.

As the simulation may be potentially very processer intensive, it is important that it will work on the machines provided by the school. Therefore a stress test of the simulation will take place to make certain that the application will be able to run at an acceptable level under normal operating conditions.

Due to the nature of the simulation, a comparison between real life biological process and those simulated will be compared. This is to check that the simulation will act as a good comparison to the real life biological system.

It is difficult for the user to enter erroneous data into the simulation due to the use of sliders for inputs. And the simulation will not generate erroneous data internally and therefore there is a lack to erroneous testing because of this.

To test that the system performs as expected it will be run at a much slower pace compared with how it will be normally operating. This is done so that system can be acutely observed.

Add more boundary and erroneous data.

Add test code + test screen shots to appendix

Trace table through ant, worker, soldier and queen classes.

Integration tests for ants is done in slow motion so process can be easily seen also run multiple times.

Mention dummy objects

Common functions e.g. die, updateGoal, doTask, draw,

# Test data

## Utilities.js - unit tests

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test | Function | Tests | Reason | Input | Expected | Result |
| 1 | validateDirection | Returns an angle in the range 0 ≤ θ < 2π | **Typical** | 5π | 1π | **PASS** |
| 2 | **Typical** | 0 | 0 | **PASS** |
| 3 | **Typical -** Negative value | -2π | 0 | **PASS** |
| 4 | angleTo | Returns the correct angle to get from coord to target | **Typical** | coord : {x : 5, y : 5}, target : {x : 7, y : 5} | ½ π | **PASS** |
| 5 | **Typical** | coord : {x : 5, y : 5}, target : {x : 7, y : 7} | 3π/4 | **PASS** |
| 6 | **Boundary –** Coordinate on top of ant i.e. navigating to its own position | coord : {x : 5, y : 5}, target : {x : 5, y : 5} | π/2 | **PASS** |
| 7 | **Typical -** wrapped direction i.e. going off map to get to position | {x : 8, y : 8}, target : {x : 1, y : 1} | 3π/4 | **FAIL** - revised and passed with function update |
| 8 | **Typical -** wrapped direction | coord : {x : 7, y : 8}, target : {x : 3, y : 2} | ≈ 3.38657 | **PASS** |
| 9 | boundary | Returns the correct warped coordinate | **Typical** | {x : 4, y : 2} | {x : 4, y : 2} | **PASS** |
| 10 | **Boundary -** Both x and y boundaries | {x : 10, y : 10} | {x : 0, y : 0} | **PASS** |
| 11 | **Typical -** X-axis boundary only | {x : 10, y : 4} | {x : 0, y : 4} | **PASS** |
| 12 | **Typical -** Y-axis boundary only | {x : 7, y : 10} | {x : 7, y : 0} | **PASS** |
| 13 | clone | Produces an exact clone of an object | **Typical** | {a : [1, 2, 3], b : {c : 4, d : '5'}, e : {f : {g : 6}}, h : [['end', 'of', ['object']]]} | {a : [1, 2, 3], b : {c : 4, d : '5'}, e : {f : {g : 6}}, h : [['end', 'of', ['object']]]} | **PASS** |
| 14 | Produces a copy of the object i.e. not by reference | **Typical** | let b = {a : 1} | {a : 1} != b | **PASS** |
| 15 | coordToIndex | Returns the correct index for a specific coordinate | **Typical** | {x : 0, y : 0} | 0 | **PASS** |
| 16 | **Typical** | {x : 5, y : 2} | 25 | **PASS** |
| 17 | distance | Returns the correct distance | **Boundary -** Both the wrapped distance and the regular distance are the same | coord1 : {x : 0, y : 0}, coord2 : {x : 0, y : 0} | 0 | **PASS** |
| 18 | **Typical** | coord1 : {x : -2, y : 1}, coord2 : {x : 1, y : 5} | 5 | **PASS** |
| 19 | getBlock | Returns a block of cells the correct size | **Boundary -** 0 size | coord : {x : 5, y : 5}, size : {width : 0, height : 0} | A single block centred at {x : 5, y : 5} | **PASS** |
| 20 | **Typical** | coord : {x : 5, y : 5}, size : {width : 3, height : 3} | A 7 wide and 7 tall block centred at {x : 5, y : 5} | **PASS** |
| 21 | **Typical -** Non square | coord : {x : 5, y : 5}, size : {width : 1, height : 3} | A 3 wide and 7 tall block centred at {x : 5, y : 5} | **PASS** |
| 22 | Returns a block of cells the correct position | **Typical** | coord : {x : 1, y : 3}, size : {width : 2, height : 2} | A 5 wide and 5 tall block centred at {x : 1, y : 3} which wraps to the other side of the map | **PASS** |
| 23 | **Boundary -** Wrapped in all quadrants | coord : {x : 0, y : 0}, size : {width : 1, height : 1} | A 3 wide and 3 tall block centred at {x : 0, y : 0} which wraps to every corner of the map | **PASS** |
| 24 | getCellCoord | Returns the correct cell for a particular coordinate | **Typical** | {x : 7.2, y : 3.6} | {x : 7, y : 3} | **FAILED -** revised and passed with function update |
| 25 | getSector | Returns sector of correct radius | **Typical** | coord : {x : 25, y : 25}, radius : 15, direction : 0, angle : 2π | A circle of radius 15 centred at {x : 25, y : 25} | **PASS** |
| 26 | **Typical** | coord : {x : 25, y : 25}, radius : 6, direction : 0, angle : 2π | A circle of radius 6 centred at {x : 25, y : 25} | **PASS** |
| 27 | Returns sector at the correct angle | **Typical** | coord : {x : 25, y : 25}, radius : 15, direction : 0, angle : π | A sector of radius 15 centred at {x : 25, y : 25} with an angle of π i.e. a semi-circle | **PASS** |
| 28 | **Typical** | coord : {x : 25, y : 25}, radius : 15, direction : 0, angle : π/4 | A sector of radius 15 centred at {x : 25, y : 25} with an angle of π /4 | **PASS** |
| 29 | Returns sector in correct direction | **Typical** | coord : {x : 25, y : 25}, radius : 15, direction : π, angle : π /4 | A sector of radius 15 centred at {x : 25, y : 25} with an angle of π/4 pointing downwards | **PASS** |
| 30 | **Typical** | coord : {x : 25, y : 25}, radius : 15, direction : 6π/ 4, angle : π /4 | A sector of radius 15 centred at {x : 25, y : 25} with an angle of π/4 pointing along the centre line of the 4th and 3rd quadrants | **PASS** |
| 31 | **Typical** | coord : {x : 25, y : 25}, radius : 15, direction : 3.657, angle : π/4 | A sector of radius 15 centred at {x : 25, y : 25} with an angle of π/4 pointing along the centre line of 3.657 radians | **PASS** |
| 32 | indexToCoord | Returns the correct coordinate from a specific index | **Typical** | 25 | {x : 5, y : 2} | **PASS** |
| 33 | **Typical** | 0 | {x : 0, y : 0} | **PASS** |
| 34 | randColour | Generates random colours. *Note*: Each is run multiple times to test that always produces valid hexadecimal colour in range #000000 to #FFFFFF. | **Typical** | N/A | A hexadecimal colour | **PASS** |
| 35 | randFloat | Returns a value within a specific range. *Note*: Each is run multiple times to test that always produces value in range. | **Typical** | {min : -1, max : 1} | In range -1 to 1 inclusive | **PASS** |
| 36 | **Typical** | {min : -0.1, max : 0.1} | in range -0.1 to 0.1 inclusive | **PASS** |
| 37 | **Typical** | {min : 0, max : 2} | In range 0 to 2 inclusive | **PASS** |
| 38 | randInt | Returns a value within a specific range. *Note*: Each is run multiple times to test that always produces value in range. | **Typical** | {min : 0, max : 100} | In range 0 to 100 inclusive | **PASS** |
| 39 | **Boundary -** No range of values | {min : 0, max : 0} | 0 | **PASS** |
| 40 | **Typical** | {min : -20, max : 20} | In range -20 to 20 inclusive | **PASS** |
| 41 | randProperty | Returns a random property from an object literal. *Note*: ran multiple times to test that not the same property each time. | **Typical** | {a : 0, b : 1, c : 2} | either 'a', 'b' or 'c' (random each time) | **PASS** |
| 42 | scaleCoord | Scales coordinates correctly | **Typical** | {x : 4, y : 2} | {x : 23, y : 12} | **PASS** |

### utilities.js unit tests – Evidence

|  |  |  |
| --- | --- | --- |
| **Test** | **Evidence** | **Comment** |
| 1 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\utilities\validateDirection - returns the correct value.PNG |  |
| 2 |  |
| 3 |  |
| 4 |  | Cells highlighted in blue is the location the ant is trying to get to. The cell in red is the location of the ant. The arrow represents the direction the ant will move in. |
| 5 |  |
| 6 |  |
| 7 |  | The image on the left is the passed test. The image on the right is the failed test. The image on the right failed as it did not point in the direction which would lead to the shortest distance between the two items. This was because it didn’t take the wrapped position into account. |
| 8 |  |  |
| 9 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\utilities\boundary - returns correct wraped coordinate 1.PNG | Cells highlighted in blue are the selected cells. |
| 10 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\utilities\boundary - returns correct wraped coordinate 2.PNG |
| 11 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\utilities\boundary - returns correct wraped coordinate 3.PNG |
| 12 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\utilities\boundary - returns correct wraped coordinate 4.PNG |
| 13 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\utilities\clone - by reference.PNG |  |
| 14 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\utilities\clone - exact clone.PNG |  |
| 15 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\utilities\indexToCoord - returns correct values.PNG |  |
| 16 |  |
| 17 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\utilities\distance - returns correct values.PNG |  |
| 18 |  |
| 19 |  | Cells highlighted in blue are the selected cells. |
| 20 |  |
| 21 |  |
| 22 |  |
| 23 |  |
| 24 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\utilities\getCellCoord - returns correct coordinate for a point 1.PNG\\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\utilities\getCellCoord - returns correct coordinate for a point - FAIL.PNG | The image on the left is the passed test while the image on the right is the failed test. Notice that the red dot represents where the exact coordinate of the ant is. While the blue square highlights the cell which the coordinate mostly lies in. |
| 25 |  | Highlighted cells in blue which are visible. The ant’s location is highlighted in red. |
| 26 |  |
| 27 |  |
| 28 |  |
| 29 |  |
| 30 |  |
| 31 |  |
| 32 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\utilities\indexToCoord - returns correct values.PNG |  |
| 33 |  |
| 34 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\utilities\randColour - Tests if generates random colours (3 times).PNG | A sample of colours shown. All colours lie in range and are different. |
| 35 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\utilities\randFloat - returns a value within a specific range.PNG | Ran multiple times to make sure rand number generated is always in the range. |
| 36 |
| 37 |
| 38 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\utilities\randInt - returns a value within a specific range.PNG | Ran multiple times to make sure rand number generated is always in the range. |
| 39 |
| 40 |
| 41 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\utilities\randProperty - returns a random property from an object literal.PNG | Ran multiple times to make sure picks a random property from the object. This is a sample of 3 runs. |
| 42 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\utilities\scaleCoord - scales correctly.PNG |  |

### utilities.js unit tests – Failed tests

#### Test 7

Test 7 failed as angleTo did not pick the angle which would lead to the shortest path to the target coordinate. The correct angle would have pointed of the map, as the wrapped distance was shorter than the normal distance. This was because angleTo did not take wrapped directions into account in its original form:

**function** angleTo(coord, target) {

**var** dx = target.x - coord.x;

**var** dy = target.y - coord.y;

**return** Math.atan2(dy, dx) + Math.PI/2;

}

The new function takes into account wrapping around the map to get the angle of the shortest path. It uses a similar algorithm to the one in the distance function.

**function** angleTo(coord, target) {

**if** (GRID\_SIZE.width **-** Math.abs(target.x **-** coord.x) **>** Math.abs(target.x **-** coord.x)) {

        dx **=** target.x **-** coord.x;

    } **else** {

        dx **=** GRID\_SIZE.width **-** (target.x **-** coord.x);

    }

**if** (GRID\_SIZE.height **-** Math.abs(target.y **-** coord.y) **>** Math.abs(target.y **-** coord.y)) {

        dy **=** target.y **-** coord.y;

    } **else** {

        dy **=** GRID\_SIZE.height **-** (target.y **-** coord.y);

    }

**return** Math.atan2(dy, dx) **+** Math.PI **/** 2;

}

#### Test 24

Test 24originally failed due to the getCellCoord function using Math.round rather than Math.floor in its operations, this meant that if the fractional part of x or y was 0.5 or greater the function would be out by one cell.

## Ant.js – unit tests

| Test | Function | Tests | Reason | Input | Expected | Result |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | Ant.scan | Adds all ants within viewing distance to ant.itemsInView.ants | **Typical** | No ants in view | this.itemsInView.ants.length = 0 i.e. cannot see any ants in view | **PASS** |
| 2 | **Typical** | Multiple ants within viewing distance | this.itemsInView.ants.length = 4 | **PASS** |
| 3 | Adds food in viewing distance to ant.itemsInView.food | **Typical** | No food in view | this.itemsInView.food.length = 0 i.e. cannot see any food in view | **PASS** |
| 4 | **Typical** | Multiple pieces of food in view | this.itemsInView.food.length = 3 | **PASS** |
| 5 | Ant.secrete | Adds pheromone of correct concentration | **Typical** | No pheromones to start | A new pheromones of concentration 0.5 | **PASS** |
| 6 | **Typical -** Adding pheromone | A pheromone of the same species | A new pheromones of concentration 0.9 i.e. 0.4 + 0.5 | **PASS** |
| 7 | **Typical -** Not adding pheromones as different species | A pheromone of a different species | A new pheromones of concentration 0.5 | **PASS** |
| 8 | **Typical -** Doesn’t exceed MAX\_PHEROMONE\_CONCENRATION | A pheromone of the same species | A new pheromones of concentration 1 | **PASS** |
| 9 | Ant.atNest | Ant is on nest or not | **Typical** | not on the nest | FALSE | **PASS** |
| 10 | **Typical** | On the nest | TRUE | **PASS** |
| 11 | Ant.seeNest | Can see nest when in range | **Typical** | Cannot see the nest | FALSE | **PASS** |
| 12 | **Typical** | Nest within view | TRUE | **PASS** |
| 13 | Ant.smell | Adds all pheromones within range to pheromonesInRange | **Typical** | No pheromones in rang | this.pheromonesInRange.length = 0 i.e. cannot smell any pheromones in view | **PASS** |
| 14 | **Typical -** Shouldn't read pheromone it’s on | 4 Pheromones in range with one on top of the ant | this.pheromonesInRange.length = 3 | **PASS** |
| 15 | Ant.takeFood | Takes correct amount of food | **Typical** | void(0) i.e. No food | 0 | **PASS** |
| 16 | **Typical** | A single piece of food of amount 3 | After three runs of Ant.takeFood food = void(0) | **FAIL -** PASSED after re writing function |
| 17 | Ant.wonder | Ant picks correct direction given some amount of pheromones | **Typical** | No pheromones in view | this.direction === this.prioritizeDirection | **PASS** |
| 18 | **Typical** | A single pheromone | Ant is pointing at the pheromone | **PASS** |
| 19 | **Typical** | Multiple pheromones | Ant is pointing slightly upwards to the right | **PASS** |

### Ant.js unit tests – Evidence

|  |  |  |
| --- | --- | --- |
| **Test** | **Evidence** | **Comment** |
| 1 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\Ant\Ant.scan - tests if an ant can see other ants.PNG |  |
| 2 |  |
| 3 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\Ant\Ant.scan - tests if an ant can see food.PNG |  |
| 4 |  |
| 5 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\Ant\Ant.secrete - tests ant drops correct pheromones.PNG |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\Ant\Ant.atNest - tests if an ant is at the nest.PNG |  |
| 10 |  |
| 11 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\Ant\Ant.seeNest - tests if an ant can see a nest.PNG |  |
| 12 |  |
| 13 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\Ant\Ant.smell - tests if ant can smell pheromones.PNG |  |
| 14 |  |
| 15 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\Ant\Ant.takeFood - test ant takes food.PNG  \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\Ant\Ant.takeFood - test ant takes food - FAIL.PNG | Top image shows both tests passing. The bottom image shows one test has failed. Due to an undefined property. See *Ant.js unit tests – Failed tests* section for more details. |
| 16 |
| 17 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\Ant\Ant.wonder - test ant follows pheromones 1.PNG | A cell highlighted in blue represents a pheromone. The red cell represents the ant. The red arrow represents the direction the ant is facing. Note that all pheromones are of the same concentration i.e. the same shade of blue. |
| 18 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\Ant\Ant.wonder - test ant follows pheromones 2.PNG |
| 19 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\Ant\Ant.wonder - test ant follows pheromones 3.PNG |

### Ant.js unit tests – Failed tests

#### Test 16

Test 16 failed originally due to an error in the Ant.takeFood function.

**if** (**this**.isFood(food))

food.removeFromMap();

There was a missing not in the stamen which meant that the food would be removed after one piece of food was taken. Here is the fixed version:

**if** (**!this**.isFood(food))

food.removeFromMap();

## Nest.js - unit tests

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test | Function | Tests | Reason | Input | Expected | Result |
| 1 | Nest.reproduce | Produces the correct probabilities | **Typical** | Soldier probability = 1, worker and queen probabilities = 0. Run 10,000 times to check probabilities. | Only soldier ants produced. | **PASS** |
| 2 | **Typical** | Soldier, Worker and Queen probabilities are all 1. Run 10,000 times to check probabilities. | A 1/3 chance of each type of ant | **PASS** |
| 3 | **Boundary** – sum will be 0 therefore dividing by 0 should produce null. | Soldier, Worker and Queen probabilities are all 0. Run 10,000 times to check probabilities. | No ants created | **PASS** |
| 4 | Nest.calcSpeciesCost | Calculates the correct sum of each characteristic cost. | **Typical** | species.chars = {speed : 0.51, stingSize : 3, eyesight : 5} | 337 | **PASS** |
| 5 | **Boundary** – No properties in CHARS | species.chars = {} i.e. no properties | 0 | **PASS** |

### Nest.js unit tests – Evidence

|  |  |  |
| --- | --- | --- |
| **Test** | **Evidence** | **Comment** |
| 1 | C:\Dropbox\projects\Ant-Simulation\writeup\assests\Maintainance\Testing\Passed Nest.reproduce test - Human anylis.PNG | This test relies on probabilities; it therefore requires human input to determine if the probabilities were close to match the theoretical result. |
| 2 |
| 3 |
| 4 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\Nest\Nest.calcSpeciesCost - correct sum.PNG |  |
| 5 |  |

## Worker.js - Integration tests

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test | Action | Tests | Reason | Input | Expected | Result |
| 1 | Searching for food. | Tests an ant will search for food. | **Typical** | A single piece of food on the map. | The ant to wonder around, scanning its surroundings and interpreting the items in view for food targets. When it spots food, its goal updates to getting food. | **PASS** |
| 2 | Retrieving food. | Tests an ant will retrieve food which it has targeted correctly. *Note*: tests canCarry and useFood functions. | **Typical** | An ant facing a single piece of food which is in viewing distance. | The ant will move towards the food until it is on top of it. The ant will determine how to best use the food i.e. eat or carry. The ant will carry out the action associated with its choice. A single piece of food should be removed from the food source and either added to the ants carrying amount or added to its health. | **PASS** |
| 3 | Dropping food. | Tests an ant will drop food to its own nest. *Note*: tests depositeFood and dropFood functions. | **Typical** | An ant carrying 3 pieces of food and a nest of the same species. | The ant to wonder around until it sees the nest. Then move directly towards the nest. Upon reaching the nest, spend 3 ticks until all food is gone. The ant’s goal will then update to finding food. All the while pheromones are being laid. | **PASS** |
| 4 | draw | Test that the ant will be drawn correctly. | **Typical** | Ant not carrying food. | Black square to be drawn. | **PASS** |
| 5 |  |  | **Typical** | Ant carrying food. | Black square with smaller square inside it. | **PASS** |

### Worker.js integration tests – Evidence

|  |  |  |
| --- | --- | --- |
| **Test** | **Evidence** | **Comment** |
| 1 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\Worker\Worker.findingFood - Worker finding Food 2.PNG\\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\Worker\Worker.findingFood - Worker finding Food.PNG | The first image shows the output of the test. It is logging events the ant is going through when searching for food.   * Searching for food means the ant is using the wondering function to walk around. * Found food target means that the ant has spotted a food source. * The updating the goal shows that the ant successfully updates its goal to the next task.   The second image shows a visual sample of the output. It is a much smaller environment. Food is green, black is the ant and blue is what the ant can see. |
| 2 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\Worker\Worker.draw - Drawn with food.PNG | This shows that the ant is carrying food as it is being drawn with a square representing it holding food. |
| 3 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\Worker\Worker.draw - Drawn with out food.png | A sample from the test showing the ant has dropped the food, as it has no longer being drawn with a square on it. *Note*: The nest is not drawn in the picture however the test creates a 1x1 mock nest on the square the ant is standing. |
| 4 | Z:\Alex On My Mac\Dropbox\projects\Ant-Simulation\app\tests\Worker\Worker.draw - Drawn with out food.png |  |
| 5 | Z:\Alex On My Mac\Dropbox\projects\Ant-Simulation\app\tests\Worker\Worker.draw - Drawn with food.PNG |  |

## Soldier.js – integration tests

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test | Action | Tests | Reason | Input | Expected | Result |
| 1 | Guarding nest | Tests an ant will guard its nests correctly. | **Typical** | A single soldier ant and its nest. | The ant will search for its nest, when it finds its nest it will move certain distance away, stop and start turning around. | **PASS** |
| 2 | Guarding food | Tests an ant will guard food correctly. | **Typical** | A single soldier ant and a single source of food. | The ant will search for the food. When the ant sees the food, it will stop and start turning around. | **PASS** |
| 3 | Attacking other ants | Tests an ant will attack ants it has targeted correctly. *Note*: tests pickTarget, follow and attack functions. | **Typical** | A single soldier ant and a single worker ant of different species. | [For solider ant only]The ant will wonder around, when it sees the other ant it will target the ant. When the ant is targeted it will begin to follow the ant until it is range to attack. The ant will then attack the other ant reducing its health until death. This will continue to happen until the other ant is dead. | **PASS** |
| 4 | Soldier.soldiersInView | Test that soldier in view returns true when there are soldiers in view. | **Typical** | No ants in view | Returns false i.e. no soldiers in view. | **PASS** |
| 5 | **Typical** | Multiple ants within viewing distance which are both workers and soldiers. | Returns true. | **PASS** |
| 6 | **Typical** | A single ant within viewing distance which is not a soldier. | Returns false. | **PASS** |
| 7 | **Typical** | A single ant within viewing distance which is a soldier. | Returns true. | **PASS** |
| 8 | Test that soldier in view correctly determines if there are friendly (i.e. of same species) ants in view. | **Typical** | A solider ant of the same species. | Returns true. | **PASS** |
| 9 | **Typical** | A soldier ant of a different species. | Returns false. | **PASS** |

### Soldier.js integration tests - Evidence

|  |  |  |
| --- | --- | --- |
| **Test** | **Evidence** | **Comment** |
| 1 | N/A | Program was stepped through in debugger and variables were watched to make sure they were producing the correct values. *Note*: test code can be found in appendix for more information on how test works. |
| 2 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\Soldier\Soldier.guardFood - Ant guards food.PNG | Green represents a single piece of food and the black square with a cross on it is an ant. |
| 3 | N/A | Program was stepped through in debugger and variables were watched to make sure they were producing the correct values. *Note*: test code can be found in appendix for more information on how test works. |
| 4 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\Soldier\Soldier.soldiersInView - test works when ants in view.PNG |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\Soldier\Soldier.soldiersInView - test works when ants in view of different species.PNG |  |
| 9 |  |

## Species.js – unit tests

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test | Function | Tests | Reason | Input | Expected | Result |
| 1 | Species.mutate | Tests that characteristics can be mutated | **Typical** | Default characteristic values | A random characteristics value is altered or the same characteristics are returned. | **PASS** |

### Species.js unit tests – Evidence

|  |  |  |
| --- | --- | --- |
| **Test** | **Evidence** | **Comment** |
| 1 | N/A | Ran mutate multiple times (as there is a chance it will not muate due to probabilisitic nature of function). |

## Food.js - unit tests

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test | Action | Tests | Reason | Input | Expected | Result |
| 1 | Food.draw | Tests the piece of food is correctly drawn for different amounts of food. | **Typical** | Amount 1 | Alpha value of 0.2 | **PASS** |
| 2 | **Typical** | Amount 5 | Alpha value of 1 | **PASS** |
| 3 | **Typical** | Amount 0 | Alpha value of 0 i.e. not visible | **PASS** |
| 4 | Food.grow | Tests that food will grow the correct amount. | **Typcial** | No food around growing food piece. | 8 new food pieces around growing block. | **PASS** |
| 5 |  | Food around growing food piece. | All food pieces amount increases by 1. | **PASS** |

### Food.js unit tests – Evidence

|  |  |  |
| --- | --- | --- |
| **Test** | **Evidence** | **Comment** |
| 1 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\Food\Food.draw - drawing piece of food correctly 1.png |  |
| 2 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\Food\Food.draw - drawing piece of food correctly 2.png |  |
| 3 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\Food\Food.draw - drawing piece of food correctly 3.png | From console output it is known that the piece of food was drawn however it has an alpha value of zero and therefore nothing to see. |
| 3 | N/A | Console output showed 6 new pieces of food where created. |
| 4 | N/A | From debug console food amount of pieces where monitered and increased by one. |

## 

## FoodSystem.js – unit tests

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test | Function | Tests | Reason | Input | Expected | Result |
| 1 | FoodSystem.addFood | Tests that food is randomly places. *Note*: Ran multiple times to check placement is random each time. | **Typical** | N/A | Random placement of food throughout the map. | **PASS** |
| 2 | FoodSystem.addFoodBlob | Tests that the blob created is of the correct size. | **Typical** | Food blob radius = 5 | A green circle radius 5 which is darkest in the middle and gets brighter the further out it goes. | **PASS** |
| 3 | **Typical** | Food blob radius = 25 | A green circle radius 25 which is darkest in the middle and gets brighter the further out it goes. | **PASS** |
| 4 | **Typical** | Food blob radius = 0 | A green circle radius 0 which is darkest in the middle and gets brighter the further out it goes. | **PASS** |

### FoodSystem.js unit tests – Evidence

|  |  |  |
| --- | --- | --- |
| **Test** | **Evidence** | **Comment** |
| 1 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\FoodSystem\FoodSystem.addFood - test adds random looking food.PNG\\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\FoodSystem\FoodSystem.addFood - test adds random looking food 3.PNG\\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\FoodSystem\FoodSystem.addFood - test adds random looking food 2.PNG | The evidence is three images showing three different random looking placements of food blobs. The test was run multiple times; these three are only a sample of the results. |
| 2 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\FoodSystem\FoodSystem.addFoodBlob - correct radius 1.PNG |  |
| 3 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\FoodSystem\FoodSystem.addFoodBlob - correct radius 2.PNG |  |
| 4 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\FoodSystem\FoodSystem.addFoodBlob - correct radius 3.PNG | The console shows that a food blob was created however was of zero size and therefore was not drawn. |

## NestPiece.js – integration tests

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test | Action | Tests | Reason | Input | Expected | Result |
| 1 | Draw | Tests that a nest piece is correctly drawn onto the map. | **Typical** | A single nest piece. | A black square. | **PASS** |

### NestPiece.js integration tests – Evidence

|  |  |  |
| --- | --- | --- |
| **Test** | **Evidence** | **Comment** |
| 1 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\NestPiece\NestPiece.draw - draws nest piece correctly.png |  |

## Pheromone.js – integration tests

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test | Action | Tests | Reason | Input | Expected | Result |
| 1 | draw | Tests that a pheromone is correctly drawn for different concentrations. | **Typical** | Amount 0.5 | Alpha value of 0.5 | **PASS** |
| 2 | **Typical** | Amount 0 | Alpha value of 0 i.e. not visible | **PASS** |
| 3 | **Typical** | Amount 1 | Alpha value of 1 | **PASS** |

Pheromone.js integration tests – Evidence

|  |  |  |
| --- | --- | --- |
| **Test** | **Evidence** | **Comment** |
| 1 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\Pheromone\Pheromone.draw - test pheromone are draw correctly 1.png |  |
| 2 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\Pheromone\Pheromone.draw - test pheromone are draw correctly 2.png | Console shows that a pheromone was drawn however its alpha value is zero. In a the real simulation, the pheromone will be instantly removed in the next tick. |
| 3 | \\fs001\01intake$\01ROBINSON\work\comp\comp4\Ant-Simulation-master(3)\Ant-Simulation-master\app\tests\Pheromone\Pheromone.draw - test pheromone are draw correctly 3.png |  |

## Queen.js – integratoin tests

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test | Action | Tests | Reason | Input | Expected | Result |
| 1 | Draw | Tests that a queen ant is correctly drawn onto the map. | **Typical** | A single queen ant. | A black circle. | **PASS** |

### Queen.js integration tests – Evidence

|  |  |  |
| --- | --- | --- |
| **Test** | **Evidence** | **Comment** |
| 1 | Z:\Alex On My Mac\Dropbox\projects\Ant-Simulation\app\tests\Queen\Queen.draw - ant is drawn correctly.png |  |

## 

## Controls.js – unit tests

*Note*: Many functions from this module are tested in *user inteface functionality testing* section.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test | Function | Tests | Reason | Input | Expected | Result |
| 1 | updateValue | Tests that correctly updates CHARS from the html value field. | **Typical** | Speed = 0.5 | Update button changes colour. Value HTML field updates. Characteristics value in CHARS updates. Ant species costs update. | **PASS** |
| 2 | **Erronous** | QueenStepsMin : 10  QueenStepsMax : 5 | Raise Error #1 i.e. queenStespMin is greater than queenStepsMax. | **PASS** |
| 3 | **Erronous** | workerFoodCost = 1000 | Raise Warning #1 i.e. worker food cost is greater than species cost. | **PASS** |
| 4 | **Erronous** | soldierFoodCost = 1000 | Raise Warning #2 i.e. soldier food cost is greater than species cost. | **PASS** |
| 5 | **Erronous** | queenFoodCost = 1000 | Raise Warning #3 i.e. queen food cost is greater than species cost. | **PASS** |
| 6 | createInputType | Test that creates a valid HTML input. | **Typical** | Characteristic : CHARS[speed]  Prop : “speed” | A new input element. | **PASS** |
| 7 | **Typical** | Non editable characteristic  Characteristic: CHARS[speed]  Prop : “speed” | A new input element with disabled attribute. | **PASS** |
| 8 | createInput | Test that creates a valid container for the input. | **Typcial** | Characteristic: CHARS[speed]  Prop : “speed” | Creates a new container for the html input. This should include a label, the input its self and a value element. | **PASS** |
| 9 | createSpeciesData | Test creates a valid HTML elements. | **Typcial** | A mock species | Creates a title for the species. Creates a colour row with label and a colour. Creates a number of ants row with label and value. A number of nests row with label and value. An Amount of food row with a label and a value. | **PASS** |

### Controls.js unit tests - Evidence

|  |  |  |
| --- | --- | --- |
| **Test** | **Evidence** | **Comment** |
| 1 | N/A | Observed console and HTML to view changes. |
| 2 | Z:\Alex On My Mac\Dropbox\Screenshots\Screenshot 2014-03-21 19.06.43.png |  |
| 3 | Z:\Alex On My Mac\Dropbox\Screenshots\Screenshot 2014-03-21 19.09.05.png |  |
| 4 | Z:\Alex On My Mac\Dropbox\Screenshots\Screenshot 2014-03-21 19.09.58.png |  |
| 5 | Z:\Alex On My Mac\Dropbox\Screenshots\Screenshot 2014-03-21 19.10.26.png |  |
| 6 | <input class="config" type="range" id="char-speed" name="speed" onchange="updateValue(this, this.value);" min="0" max="0.2" step="0.01"> | This is the HTML generated by the function. |
| 7 | <input class="config" type="range" id="char-speed" name="speed" onchange="updateValue(this, this.value);" disabled="disabled" min="0" max="0.2" step="0.01"> |
| 8 | <tr class="config" title="The speed that an ant can move"><td class="config">Speed</td><td class="config" id="input-container"><input class="config" type="range" id="char-speed" name="speed" onchange="updateValue(this, this.value);" min="0" max="0.2" step="0.01"></td><td class="config" id="char-speed-value">0.04</td></tr> |
| 9 | <table class="species 1"><tr class="species 1" id="1-label-row" style="color: rgb(28, 28, 28);"><td class="species 1" onclick="select(this)">Species: 1</td><td class="species 1 toggleVisibility" onclick="toggleClassVisibility(this)">-</td></tr><tr class="species 1" style="display: table-row;"><td class="species 1">Colour</td><td class="species 1" id="1-colour-data" style="background-color: rgb(28, 28, 28);"></td></tr><tr class="species 1" style="display: table-row;"><td class="species 1">Number of ants</td><td class="species 1" id="1-antNum-data">1</td></tr><tr class="species 1" style="display: table-row;"><td class="species 1">Number of Nests</td><td class="species 1" id="1-nestNum-data">0</td></tr><tr class="species 1" style="display: table-row;"><td class="species 1">Amount of food</td><td class="species 1" id="1-foodAmount-data">0.00</td></tr></table>  Z:\Alex On My Mac\Dropbox\Screenshots\Screenshot 2014-03-21 21.13.57.png |

## map.js – unit tests

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test | Action | Tests | Reason | Input | Expected | Result |
| 1 | drawGrid | Tests that the grid is correctly drawn. | **Typical** | GRID\_SIZE = {width: 10, height: 10}  CELL\_SIZE = {width: 10, height : 10} | A 10x10 grid | **PASS** |

|  |  |  |
| --- | --- | --- |
| **Test** | **Evidence** | **Comment** |
| 1 | Z:\Alex On My Mac\Dropbox\projects\Ant-Simulation\app\tests\Food\Food.draw - drawing piece of food correctly 3.png |  |

## Compatibility tests

The tool has been tested on multiple web browsers on multiple operating systems to check its compatibility. This is necessary as the tool may be set for homework and the pupils need to be able to use it at home.

**Note: The canvas HTML 5 element which is used to display the simulation is the same in all browsers which support it for its main features. The use of the canvas element in the simulation is specifically designed to only use the core features of the canvas so that support can be guaranteed by browsers which have an implementation of the canvas element.**

| Test | Description | Reason | Evidence | Result |
| --- | --- | --- | --- | --- |
| 1 | Chrome – Windows 8 | Chrome being the most popular browser and windows 8 being a popular operating systems. | C:\Dropbox\projects\Ant-Simulation\app\tests\Browser compatibility\win8_chrome_27.0.png | **PASS** |
| 2 | Firefox – Windows 8 | Firefox is also a popular web browser and windows 8 being a popular operating systems. | C:\Dropbox\projects\Ant-Simulation\app\tests\Browser compatibility\win8_firefox_20.0.png | **PASS –** The slider input html element is not supported in Firefox and so the inputs are text boxes rather than sliders. |
| 3 | Safari – IOS (iPad) | Safari is the default browser on IOS and so properly most used. The iPad is one of the only portable devices which the simulation can run on smoothly, phones are not powerful enough. | C:\Dropbox\projects\Ant-Simulation\app\tests\Browser compatibility\ios_iPad-3rd_5.1_portrait.jpg | **PASS** |
| 4 | Chrome - Ubuntu | Chrome being the most popular browser and Ubuntu being the most popular Linux distribution this test is likely to catch the small number of people who use a Linux OS. | C:\Users\0x\Downloads\140203-162615-begly.github.io-9871292\140203-165738-chrome-32.0.1700.102-ubuntu-12.04-lts-41d660b1228e545dbe8d57c4dfd6b8d1.png | **PASS** |
| 5 | Safari – OS X | Safari is the default browser on OS X and therefore probably one of the most popular. | C:\Users\0x\Downloads\140203-162615-begly.github.io-9871292\140203-165115-safari-5.1.7-mac-os-x-10.7-8c5186b2a1e265bb6967357aab5963fc.png | **PASS** |

## Stress tests

The simulation went through a series of stress tests to see how well it copes with large numbers of ants. Ants where picked as the independent variable as the number of ants is the primary contribution to the time complexity of the algorithm. Only worker ants are used as they act as a good average complexity (as queen ants require less processing then worker ants and soldier ants require more processing then worker ants).

#### Setup

The simulation was set with a GRID\_SIZE of 250x250. The simulation was slightly modified so that an exact number of ants would be created at random positions at the start of the simulation rather than having to wait for multiple nests to produce enough ants.

The simulation was run on chrome 32 on the windows 7 operating system. The hardware was a school computer similar to the ones used in the biology department and thus a good representation of the performance to be expected when the simulation is used.

#### Frame rate and usability

The frame rate in the simulation will not be noticeable until it gets extremely low e.g. < 2 fps. This is because the frame rate is the same as the tick time, thus meaning higher frame rates will mean the simulation runs faster and lower frame rates means the simulation runs slower. This is done to increase performance of the simulation so that it can run as fast as possible so the user can see how the change of characteristics affect the ant’s behaviour faster.

#### Results

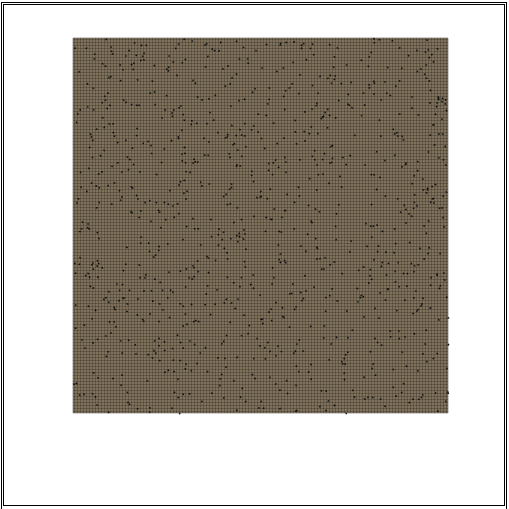
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Tests | Reason | Expected | Result |
| 1 | 1 ant | The starting number of ants on the simulation. This is certain to happen in every simulation. | Very smooth frame rate (> 60fps) | **PASS** |
| 2 | 10 ants | A small nest of ants (< 20 ants per nest). This will very nearly happen every simulation run. | Very smooth frame rate (> 60fps) | **PASS** |
| 3 | 100 ants | Multiple medially sized nests (> 20 ants per nest). This will happen in simulations left on few about 2 minutes | Smooth frame rate (> 30fps) | **PASS** |
| 4 | 1000 ants | Multiple large nests (> 80 ants per nest). This is not likely to happen to most users. | Useable frame rate (> 5fps) | **PASS** |
| 5 | 10000 ants | Multiple very large nests (> 200 ants per nest). This may not happen atoll in most simulations, no matter how long they are left on for due to the characteristics chosen and either ants dying out due to being out competed for food or lack of food. | Very choppy frame rate (> 0.25fps) | **PASS** |
| 6 | 100000 ants | Many multiples of extremely large colonies of ants (> 5000 ants per nest). This is extremely unlikely to occur as it would require a significant amount of time to get the simulation to this point as well as an extremely large map and large food source. | Unusable (< 0.25fps) | **PASS** |

#### C:\Dropbox\projects\Ant-Simulation\app\tests\stress\1 ant.PNGC:\Dropbox\projects\Ant-Simulation\app\tests\stress\10 ants.PNGC:\Dropbox\projects\Ant-Simulation\app\tests\stress\100 ants.PNGStress tests – Evidence

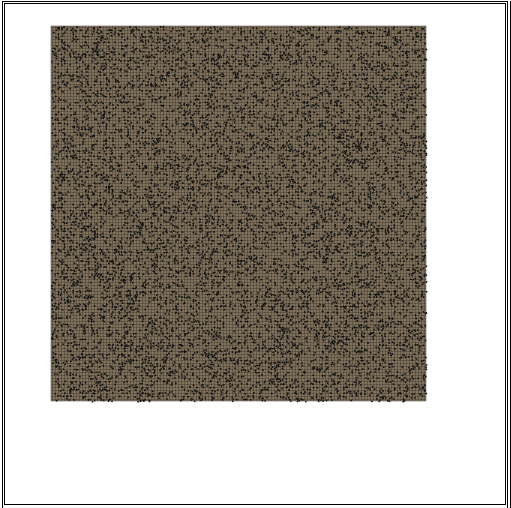
Test - 1 ant (200 fps)

Test - 10 ants (100 fps)

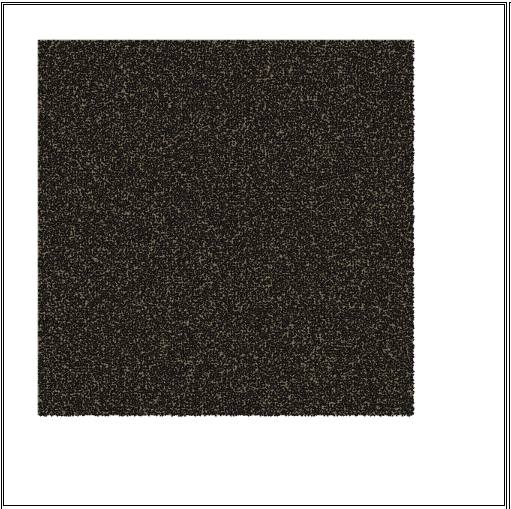
Test - 100 ants (40 fps)



Test - 1000 ants (8 fps)



Test - 10000 ants (1 fps)

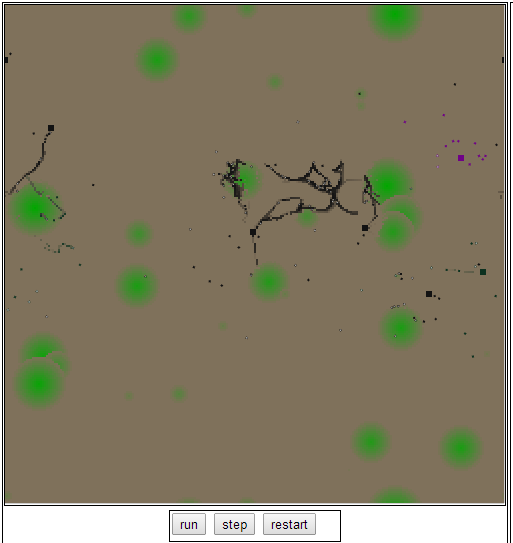


Test - 100000 ants (N/A fps)

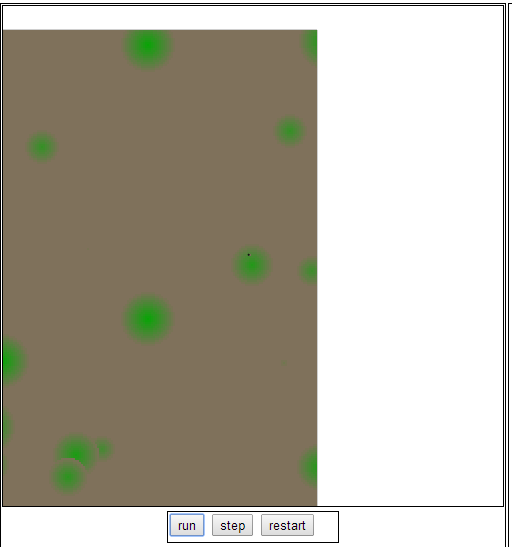
## User interface functional testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Tests | Method | Expected | Result |
| 1 | The run button starts and stops the simulation. | Press the pause button and then press the run button. | When pause button pressed simulation stops and the pause button will change to a run button. When the run button is pressed the simulation starts from the place it left off and the run button changes back into a pause button. | **PASS** |
| 2 | Press the space bar. | **PASS** |
| 3 | The step button steps the simulation forwarded a single tick. | Press the step button. | All ants to go through a single update cycle i.e. do a single action such as pick up a piece of food or attack an ant. All ants (apart from those collecting food) will move an amount equal to their species speed characteristic. | **PASS** |
| 4 | Press the “s” key. | **PASS** |
| 5 | The restart button restarts the simulation. | Press the restart button. | The simulation restarts, keeping all characteristics for the starting species the same. | **PASS** |
| 6 | Press the “r” key. | **PASS** |
| 7 | Species data selection. | Click on the species from the species panel. (Selecting species 46) | Map to centre on the first nest in the spices. The selected species to be highlighted in the species data panel. The characteristics of the selected species to appear in the characterises panel. | **PASS** |
| 8 | Expanding species data | Press the + button of species 87 | Data about the species expands showing the species colour, number of ants, number of nests and amount of food. | **PASS** |
| 9 | Contracting species data | Press the – button of species 87 | Data about the species colour, number of ants, number of nests and amount of food to be hidden. | **PASS** |
| 10 | Pan around simulation | Click and drag mouse on simulation. | Simulation to move under the mouse until mouse is released. | **PASS** |
| 11 | Press arrow keys. | **PASS** |
| 12 | Zoom into simulation | Scroll mouse wheel. | Simulation to either zoom or zoom out depending on direction of scroll/+ or – key. | **PASS** |
| 13 | Press + and – keys. | **PASS** |
| 14 | The random button randomizes the selected species characteristics. | Pressing the random button. | The characteristics of the selected species to be changed to random values. The update button to highlight. And the ants health to change. | **PASS** |
| 15 | The reset button resets selected species characteristics to default values. | Press the reset button. | The characteristics of the selected species to change to the default values. The update button to highlight. And the ants health to change. | **PASS** |
| 16 | The update button pushes the changed value to the simulation. | Press the update button. | The update button to return to its normal colour. The species in the simulation value to be updated. | **PASS** |

Test 5

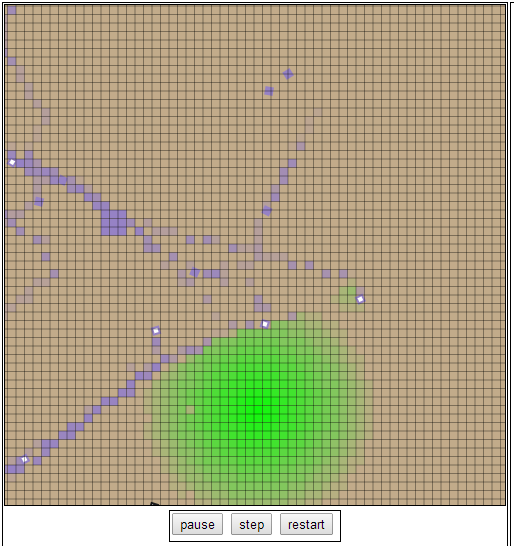
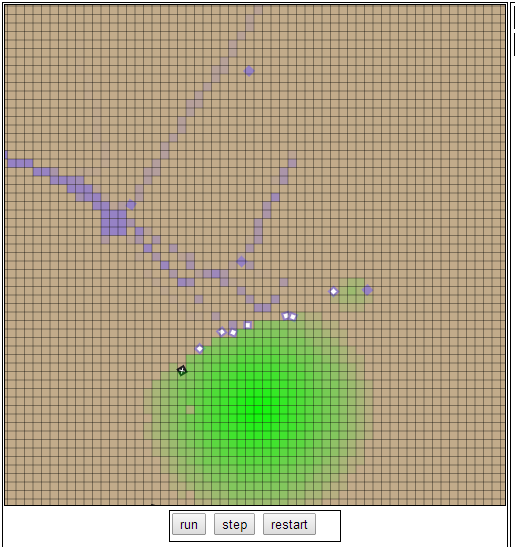


Pre restart



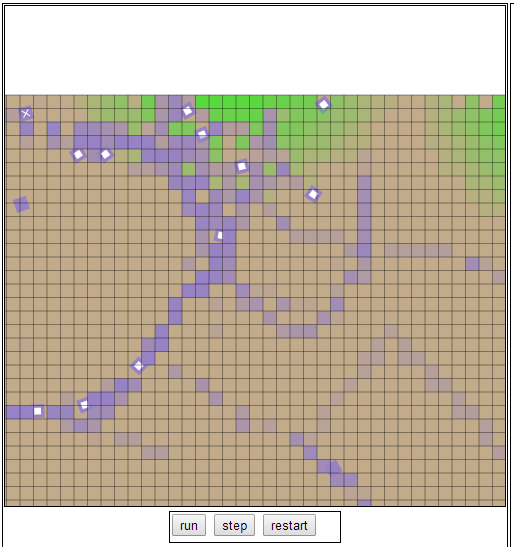
Post restart

Test 1

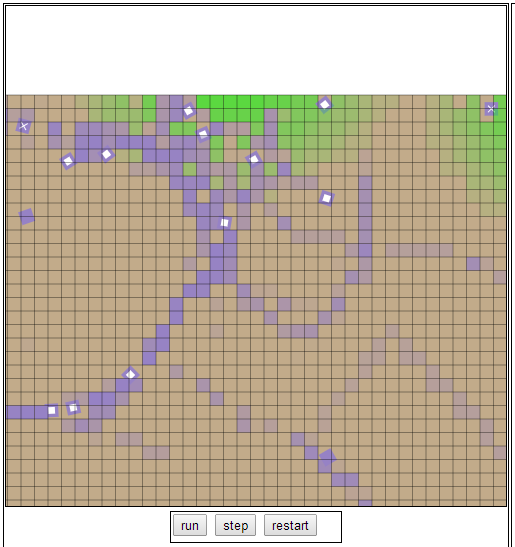


Pause pressed

Run pressed



Pre step



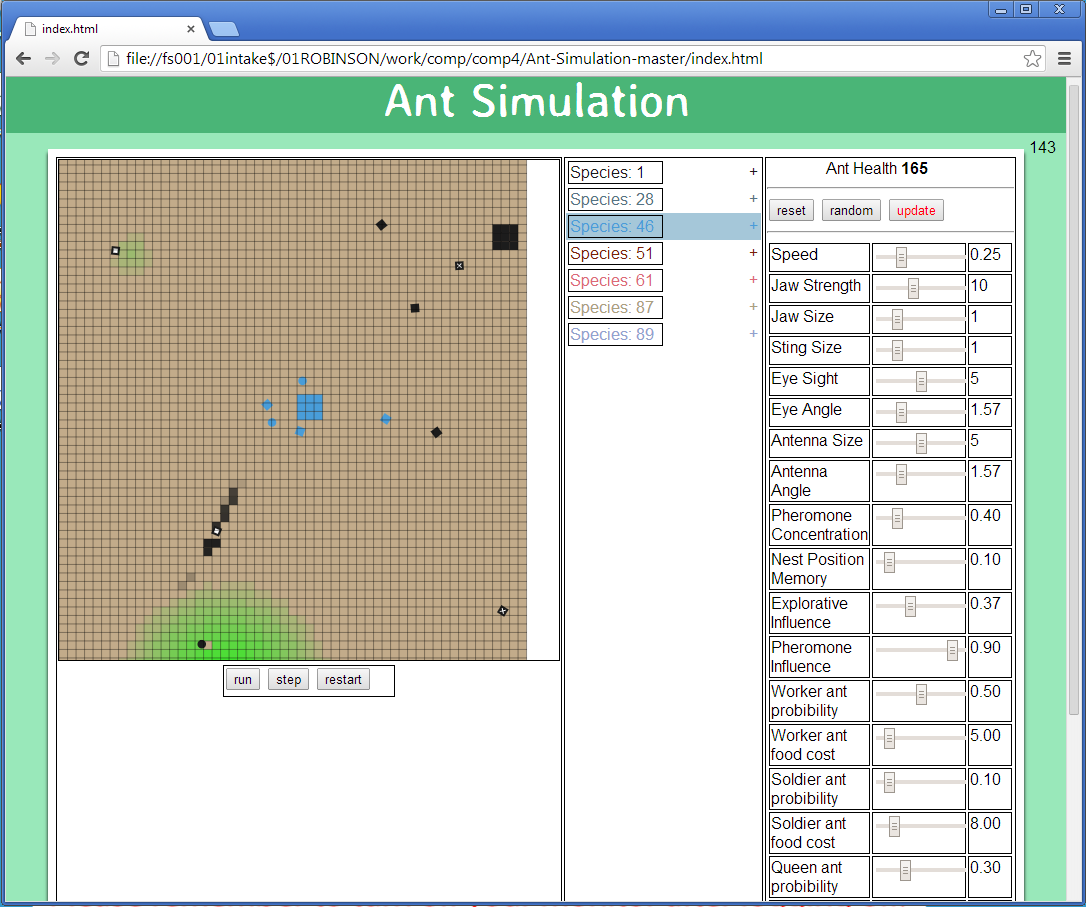
Post step

Test 3

Test

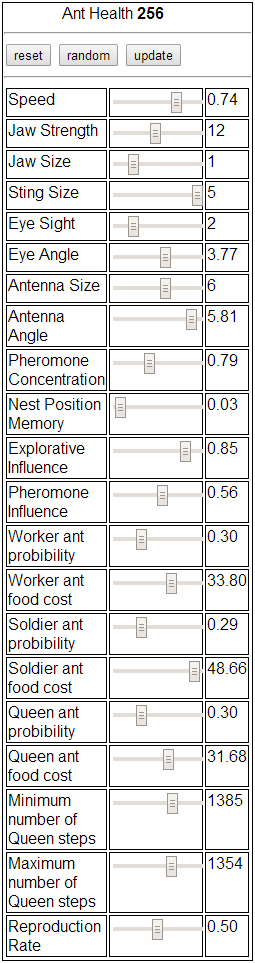


Pre selection



Post selection of species 46

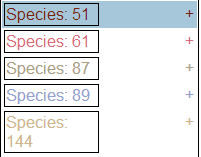
Test 14



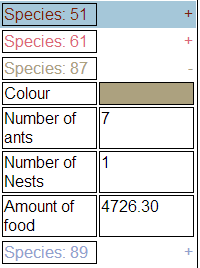
Pre random



Post random



Test 9



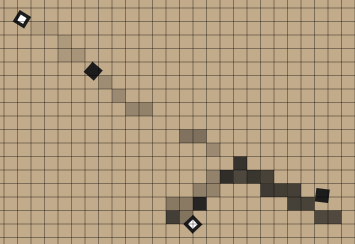
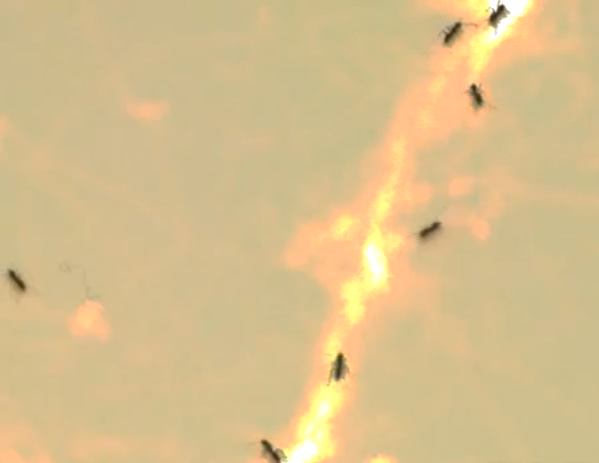
Test 8

## Biological tests

The simulation is based on ants, the behaviour of the simulated ants is changed by altering a species characteristics. This behaviour can sometimes closely resemble real ants however it can be equally a long way of how real ant behaves. For example by changing the antenna size characteristic to 0 will mean that ants can no longer smell pheromones. The behaviour of ants in this state will not model the behaviour of real ants (which can smell pheromones). This means that in order to test the biological trueness of the simulation would be to find the characteristics in the simulation which most closely match a real ant; this is very difficult as the simulation only has a limited range of characteristics to choose from while real ants have a much more complex system of genes.

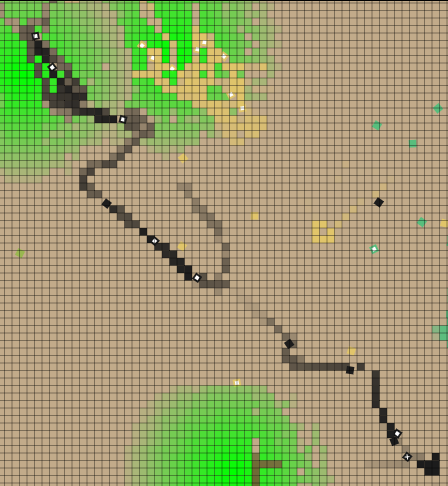
A comparison between real ants and the simulated ants will be used to compare how well the simulated ants model the real ants. Also approximations of behaviour will also be compared between the two.

### Pheromones

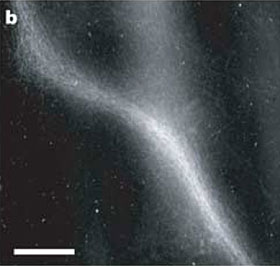
Pheromones laid by real ants are most concentrated where most ants have been as they cumulate. This is seen in the picture of real ants by the brighter yellow colour. This can be seen in the simulation by a darker colour pheromone. In videos, ants will tend to head in the direction which has the strongest concentration of pheromones. Again this is mirrored in the simulation (See tests 17, 18 and 19 in Ant.js unit tests). Real ants will also sometimes leave the trial to go off looking in a new direction. This is shown in the diagram of the simulation where the top right ants did not follow the stronger pheromone trial left by the ant in the centre left. And in the picture of real ants by the ant on the far left of the screen who left the trial in search for food. Finally pheromones in real life evaporate over time, this is also modelled in the simulation as pheromones concentration decrease as time increases.

### Searching for food

Real ants walk randomly until an ant finds food, it then navigates back to the nest (depending on type of ant either following a pheromone, memory or randomly). Other ants then follow the trial left by the first ant to find the food, reinforcing the strength of the trial making it more popular. This can all also be seen in the simulated ants.

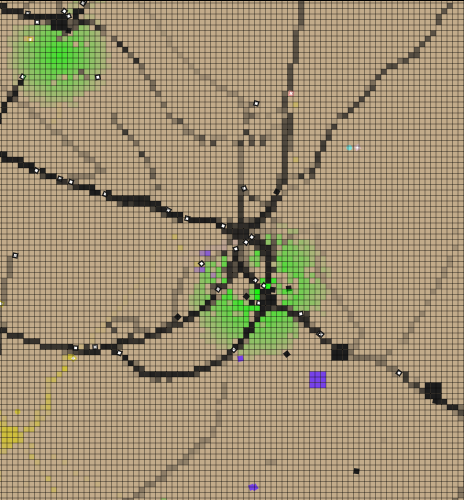
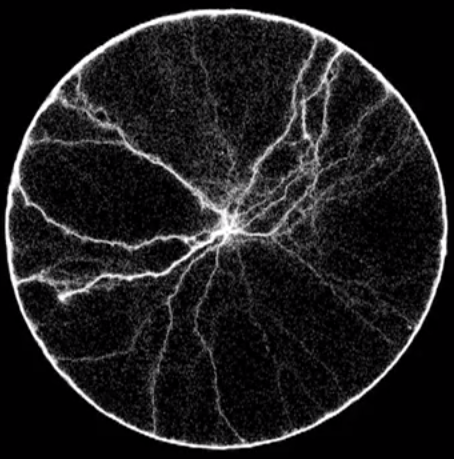
The screen shot shows a trial which leads from the nest to food. A similar looking trial is seen in the picture of real ants leading from the nest to food.

NEST



FOOD

### Trial patterns

Real ant trials come into the nest from all directions, the main trials (strongest concentration pheromones) are roughly 70 degrees apart (to maximise spread to new food sources). This is very similar to the simulation which has trials from all directions and the main trials are separated by about 70 degrees also. The concentration of pheromones is also similar, although the scale of the simulation and the real pheromone trials of ants are different the pattern of most concentrated pheromones around the nest with roughly straight thin lines of high concentration pheromones leading off in multiple directions as well as a small number of considerable weaker pheromones from all directions can be seen in the real pheromones and the simulated.